

Beam Summing

Local application

Fri, Jun 17, 1994

For monitoring long term accumulations of raw data such as beam charge, it is necessary to do the pulse-by-pulse summing locally, then making the accumulations available to any host system. This note describes an implementation for this as a local application.

Parameters layout:

```
ENABLE  B<00C4> BSUM ENABLE
BEAM    B<009F> NO BEAM STATUS
OUTADDR <2F00>
CHAN1   C<001B>   BEAM ACCUM TEST
CHAN2   C<0000>
CHAN3   C<0000>
CHAN4   C<0000>
```

The ENABLE Bit# enables operation of the BSUM local application. The BEAM status Bit# indicates what bit signals the presence of a scheduled beam pulse of the type summed by this application. The “beam” state is the sign bit of this parameter. The OUTADDR parameter is a 16-bit address in low memory where the output results are written. This address should be in an area that is zeroed at system reset time, in order to start the accumulations at zero and to signal a reset has occurred. (Consult with an expert to determine an appropriate address to use.) The rest of the parameters are analog channel#s whose readings provide the raw data to be summed. A zero channel# is ignored, but it occupies a “slot” in the output data structure.

The data structure of the accumulated data is as follows:

```
sum: ARRAY[1..4] OF Integer; { 4-word sum of beam data }
cnt: ARRAY[1..2] OF Integer; { 2-word sum of beam cycles }
tot: ARRAY[1..2] OF Integer; { 2-word sum of all cycles }
```

(Here, an “Integer” is a 16-bit word.) For each channel specified, a “slot” of 8 words is used starting at OUTADDR. The SSDN of the Acnet database entry for the reading property can include this OUTADDR. For example, if the source node were node 61E, and the base address of the data structure were 00002F00, one could then have the following SSDN structure:

1D02/061E/0000/2F00.

Upon reading this data, a host program can compute the accumulation (as a double precision floating point value) of each signal as follows, where $k=32768$:

```
acc:= ((sum[1]*k + sum[2])*k + sum[3])*k + sum[4];  
nBP:= cnt[1]*k + cnt[2];  
all:= tot[1]*k + tot[2];
```

These values need to be referenced to the set of values obtained during the last query by the host program, in order to get the amount of beam that has been accumulated over the last query interval. Upon conversion to engineering units, the data can be archived as needed. Note that this scheme works for multiple users without conflict.

The front end keeps 15 bits of precision in each word in order to maintain positive values that simplify the above formulas in hi-level language. (It also avoids the word-swap problems that can result from differences with Vax data formats, since words are automatically byte-swapped by Fermilab networking hardware.) Note that a 60-bit long summation will never overflow in anyone's lifetime. Also, 30 bits of pulses at 15 Hz is more than 2 years. Site-wide power outages occur more often than that. A reset of the front end clears the accumulation area.

The host program will zero its own version of the accumulations according to its particular implementation. If the host program finds that the accumulation has dropped to a lower value than it had during its previous query, it can assume that the system has reset and restarted its accumulations. The most beam accumulation that could have been lost would be that which occurred since the last query. Assuming a one-minute query interval, for example, this should not be significant. (Typical times between resets of Linac front end stations are measured in months.)